Reinforcement Learning

Chapter 5. Monte Carlo Methods

Presented by:

Yurou He and Ruqing Xu

March 6, 2025

Monte Carlo

- The Monte Carlo Method is named after the Monte Carlo Casino in Monaco.
- It relies on randomness and probability, much like gambling games such as dice and cards in the casino.

Monte Carlo Methods

- Monte Carlo methods are computational techniques that use *repeated random sampling* to approximate numerical solutions to complex problems.
- Commonly applied in physics, finance, artificial intelligence, and statistics.
- Key Applications:
  - Estimating π by randomly sampling points within a unit square.
  - Approximating high-dimensional integrals using Markov Chain Monte Carlo (MCMC).
  - Evaluating state-action values in reinforcement learning.

Monte Carlo Methods

Example: Estimating π

- We want to approximate mathematical constants π by knowing

$$\text{Circle area} = \pi r^2$$

- Given a square of side length 2 enclosing a unit circle, the ratio of points inside the circle to the total points gives an estimate of π:

$$\pi \approx 4 \times \frac{\text{Number of points inside the circle}}{\text{Total number of points}}$$

- Generate N random points (x_i, y_i) inside the square and check:

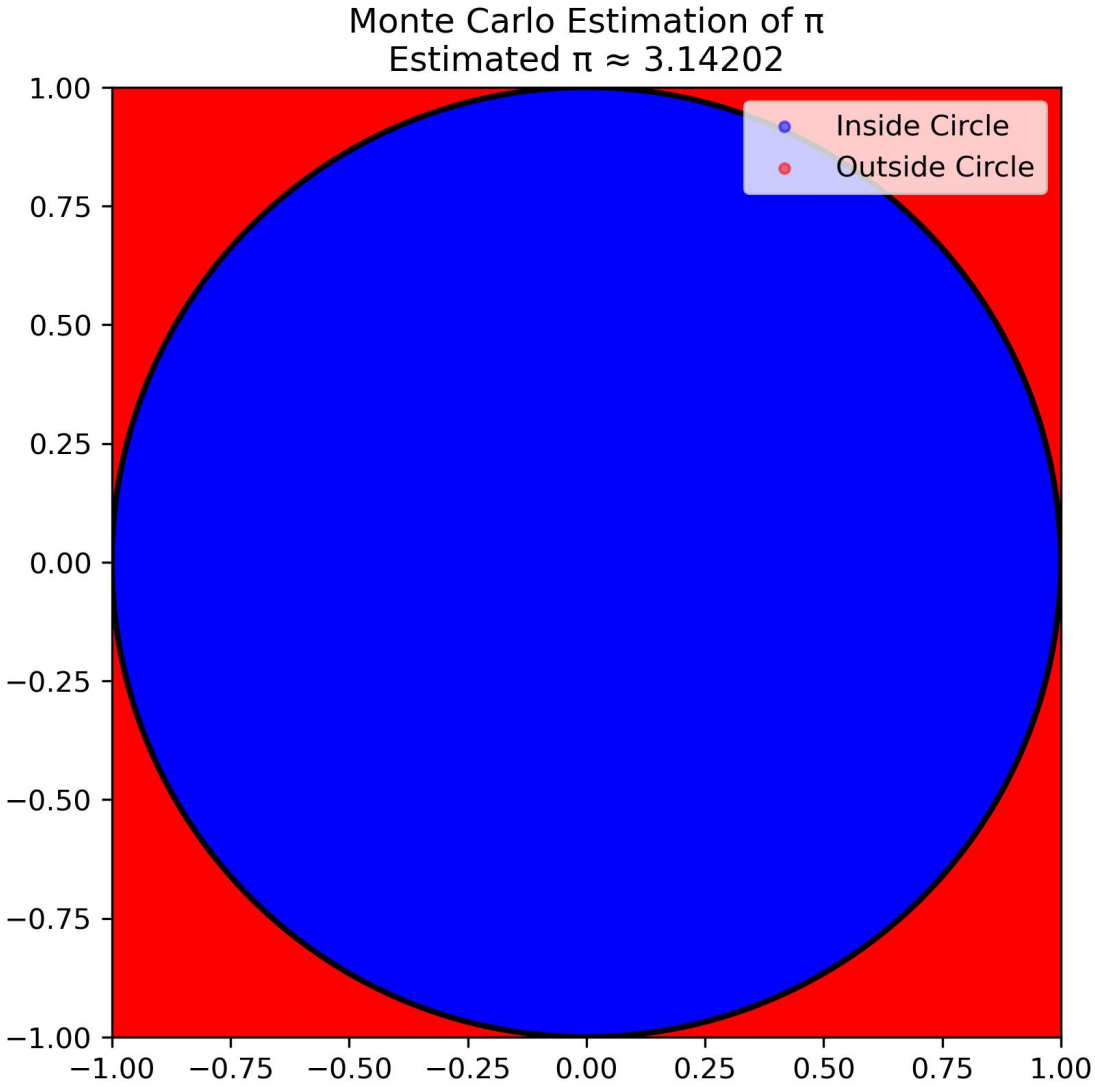
$$x_i^2 + y_i^2 \leq 1, \quad \text{for all } i \in \{1, 2, \ldots, N\}.$$

- Compute the ratio of inside points to total points:

$$\pi \approx 4 \times \frac{\text{Points inside}}{N}$$

Monte Carlo

Example: Estimating π



Monte Carlo in Reinforcement Learning

Unlike dynamic programming (DP), which relies on knowing the environment's **transition dynamics**, MC learns from **sampled experiences** (episodes) by averaging observed returns.

- No need for an explicit environment model (model-free).
- Used for policy evaluation and policy improvement.
- Advantages:
  - No need for transition probability knowledge.
  - Simple and applicable to large-scale problems.
- Disadvantages:
  - Requires complete episodes (not suited for continuing tasks).
  - Can have high variance and slow convergence.
  - Some algorithms do not have proven theoretical guarantee.

Example: Blackjack

Game Rules

- A player competes against the dealer.
- The game starts with two cards dealt to both of them:
  - Aces can count as either 1 or 11 (count as 11 if sum ≤ 21).
  - Face cards (King, Queen, Jack) count as 10.
  - Numbered cards retain their value.
- The player's two cards are hidden, while the dealer reveals one card.
- The goal is to obtain cards whose total value is as close as possible to 21, without exceeding it.

Example: Blackjack

Game Rules

- Player's Action:
  - If the player has 21 from the initial two cards, it is called a natural.
  - A natural wins immediately unless the dealer also has a natural (which results in a draw).
  - If the player does not have a natural, they can:
    - Hit: draw additional *cards* until reaching 21 or busting.
    - Stick: end their turn, allowing the dealer to play.
  - If the player's sum exceeds 21, they go bust and lose.
- Dealer's Action:
  - The dealer follows a fixed strategy:
    - Hit if the total sum is less than 17.
    - Stick if the total sum is 17 or greater.
  - If the dealer goes bust, the player wins.

Example: Blackjack

Limitations of DP in Blackjack

- DP requires knowing the full transition dynamics $p(s', r|s, a)$, which is difficult in games like blackjack.
  - This requires computing probabilities of all possible dealer outcomes before applying DP.
- Even if we know $p(s', r|s, a)$, computing state transitions can still be complex.
  - For example, if the player has 14 and chooses to stick, the probability of winning depends on the dealer's actions.
- However, Monte Carlo (MC) methods can
  - Learns from sampled episodes (experiences) without knowing transition probabilities.
  - Even when the environment is fully known, MC avoids complex probability calculations.

MC Prediction of $V_\pi(s)$

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Backwards !

Cumulative rewards counting back from T to t

Returns(St) update at most once in each episode

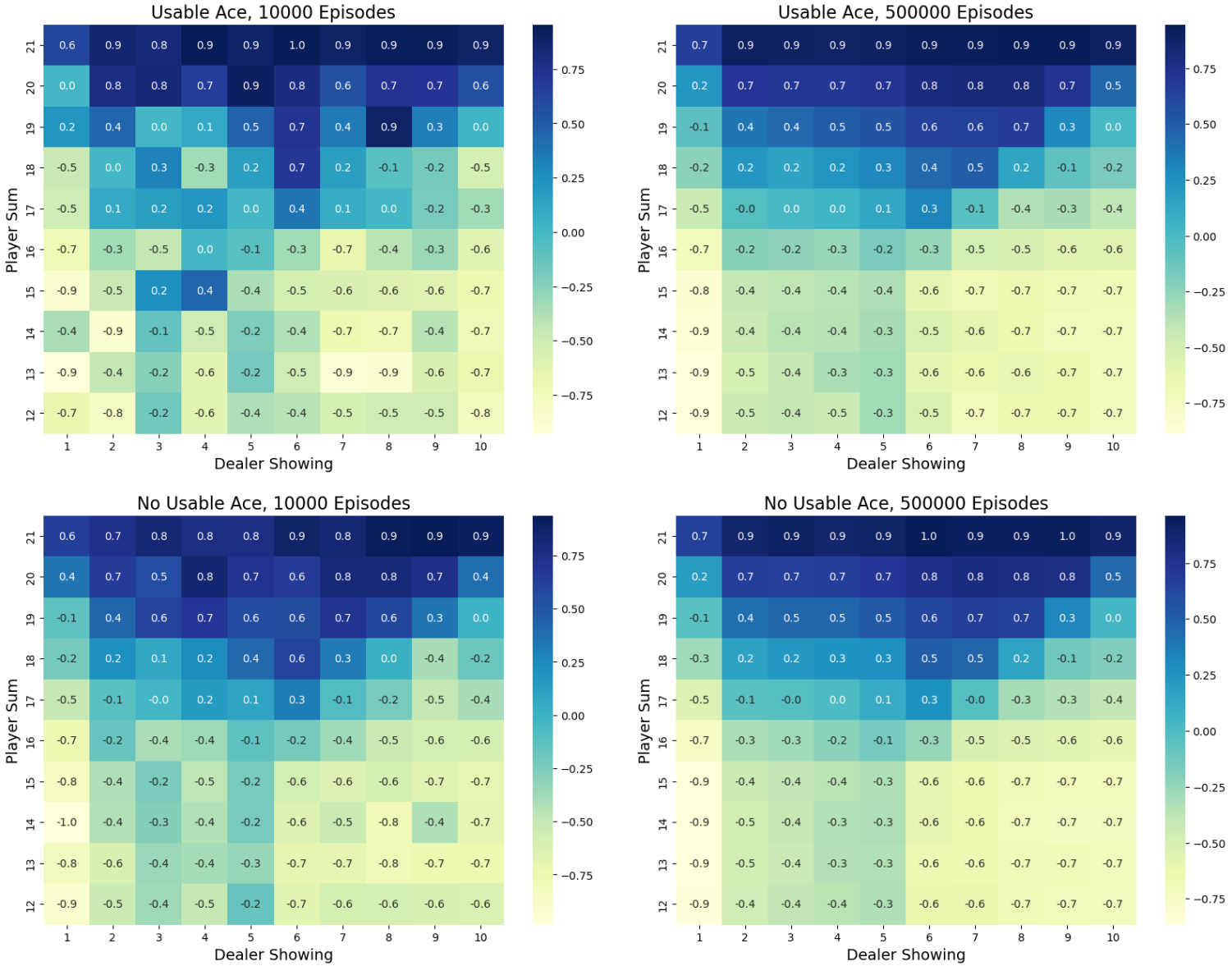
Example: Blackjack

Blackjack as an Episodic Finite MDP

- Each game is an **episode** (finite-horizon MDP).
- The **state** is defined by three variables (total of 200 states):
  - Dealer's showing card (1–10, Ace).
  - Player's current sum (12–21).
  - Whether the player has a *usable* Ace (Yes/No).
- The player makes decisions based on their current state, with two possible **actions**: hit or stick.
- The dealer follows a fixed, non-adaptive strategy.
- Cards are drawn from an infinite deck and no discount.
- **Rewards**: +1 for a win, -1 for a loss, 0 for a draw.

Example: Blackjack

Monte Carlo Evaluation



The same methods for $Q_\pi(s, a)$?

- Since the model is unknown, we need $Q_\pi(s, a)$ to derive a policy.
- How about using the same methods (first visit) to estimate the value starting from any state-action pair (s, a)?
- Problem: some state-action pair may never be visited by π (e.g. deterministic policy).
- Need to *maintain exploration*.

Question: why is it not a problem for learning $V_\pi(s)$? What is implicitly assumed?

Sol. 1: Exploring Starts

- Just force it!
- Specify that each episode starts in a (s, a) pair and $\Pr(s, a) > 0, \forall s, a$.

MC Control with Exploring Starts

Estimating Q_π and π^*

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode): Exploring starts

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$: (Backwards)

$G \leftarrow \gamma G + R_{t+1}$ → Cumulative return from T back to t

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

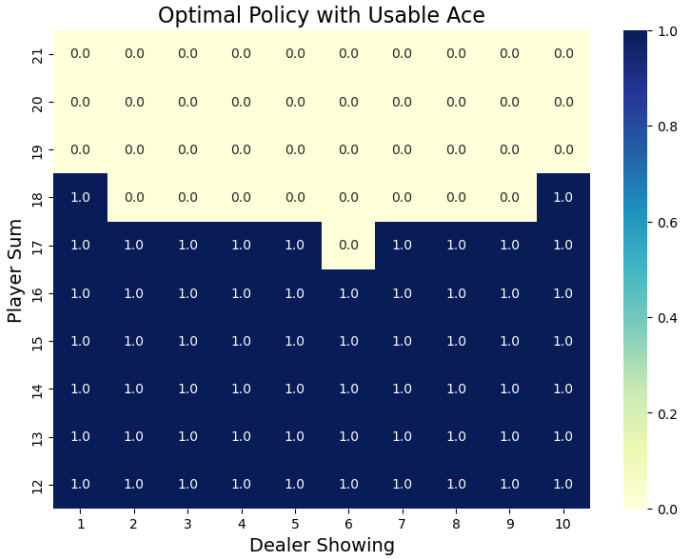
Append G to $Returns(S_t, A_t)$

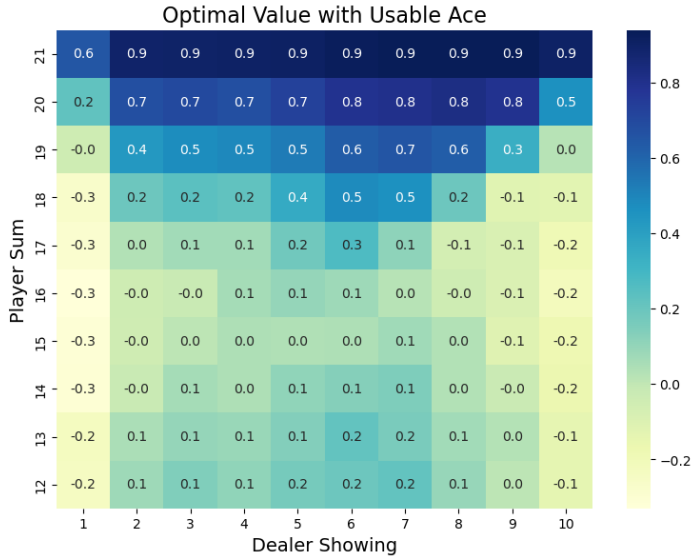
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

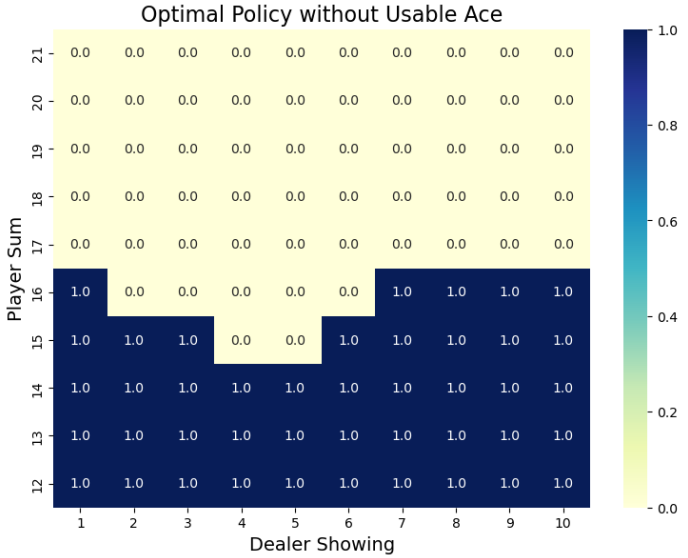
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ Deterministic, greedy

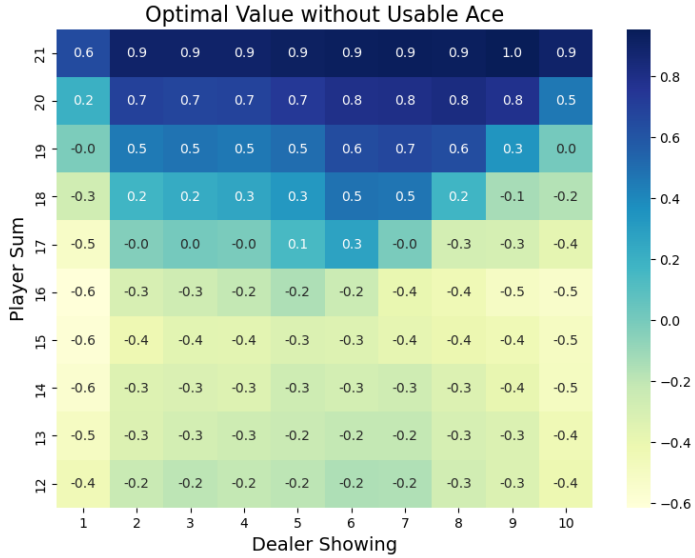
Example: Blackjack

Monte Carlo Control with Exploring Starts









Sol. 2: Restricting to ϵ-soft policies

Definition (ϵ-soft policies)

Policies for which $\pi(a \mid s) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$ for all states and actions.

Definition (ϵ-greedy policies)

Policies for which all non-greedy actions are given probability $\frac{\epsilon}{|\mathcal{A}(s)|}$ and the greedy action is given the remaining probability.

- We can optimize only in the ϵ-soft policy space, and the resulting policy will be optimal among the ϵ-soft policies.

MC control with ϵ-soft policies

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Same

Sol. 3: Off-policy Prediction via Importance Sampling

- ϵ-soft policies learning is a compromise because the value functions learned are for the optimal ϵ-soft policy, not the true optimal.
- How can we learn about the optimal policy while behave like an exploratory policy?
- Separate the two!
- Learns the *target policy* from data generated by a *behavioral policy*.
- Use *importance sampling*: weighting returns by the likelihood of trajectories occurring under the target and behavior policies.

Importance Sampling

$$\Pr(A_t, S_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi) = \prod_{k=t}^{T-1} \pi(A_k \mid S_k)\, p(S_{k+1} \mid S_k, A_k)$$

Importance sampling ratio:

$$\rho_{t:T-1} = \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k)\, p(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k \mid S_k)\, p(S_{k+1} \mid S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}.$$

Ordinary importance sampling:

$$V(s) := \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}\, G_t}{|\mathcal{T}(s)|}.$$

Weighted importance sampling:

$$V(s) := \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}\, G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}.$$

MC prediction with importance sampling

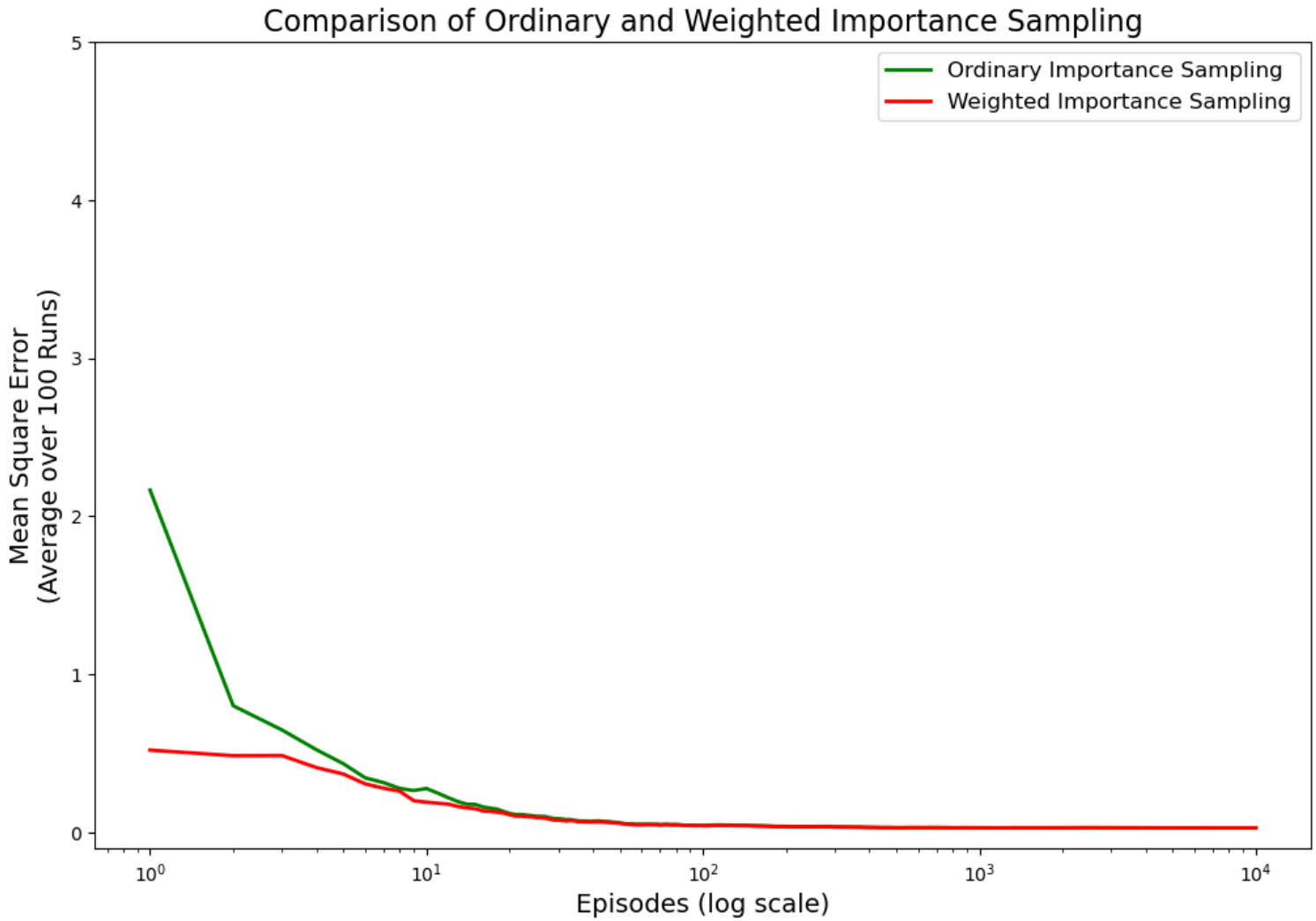
Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$ # incremental way to calculate for W

Example: Blackjack

Off-policy Monte Carlo Prediction of Q_π

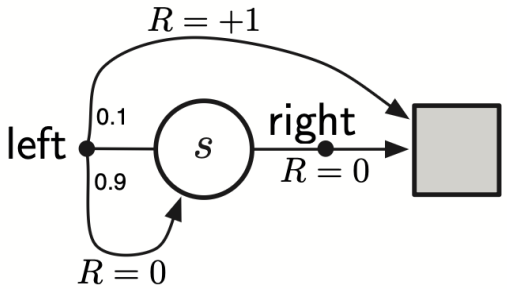


Ordinary vs. Weighted Importance Sampling

- Ordinary importance sampling in off-policy learning can lead to infinite variance.
- This happens when the importance sampling ratio ρ grows exponentially.
  - The behavior policy (μ) and target policy (π) are significantly different.
  - Some trajectories have extremely large importance weights, causing instability.

Example: Infinite Variance

The One-State MDP Setup



$$\pi(\text{left}|s) = 1$$

$$b(\text{left}|s) = \frac{1}{2}$$

- The environment has one nonterminal state s.
- Two actions:
  - Right (a_R): Immediate termination with reward $R = 0$.
  - Left (a_L): Returns to s with probability 0.9, terminates with $R = 1$ otherwise.
- The target policy π always selects Left (a_L).
- The behavior policy b chooses Left (a_L) and Right (a_R) with equal probability (0.5 each).

Example: Infinite Variance

Importance Sampling Ratios

- The ratio $\rho_{t:T(t)-1}$ transforms the observed return G_t from behavioral policy to have the right expected value:

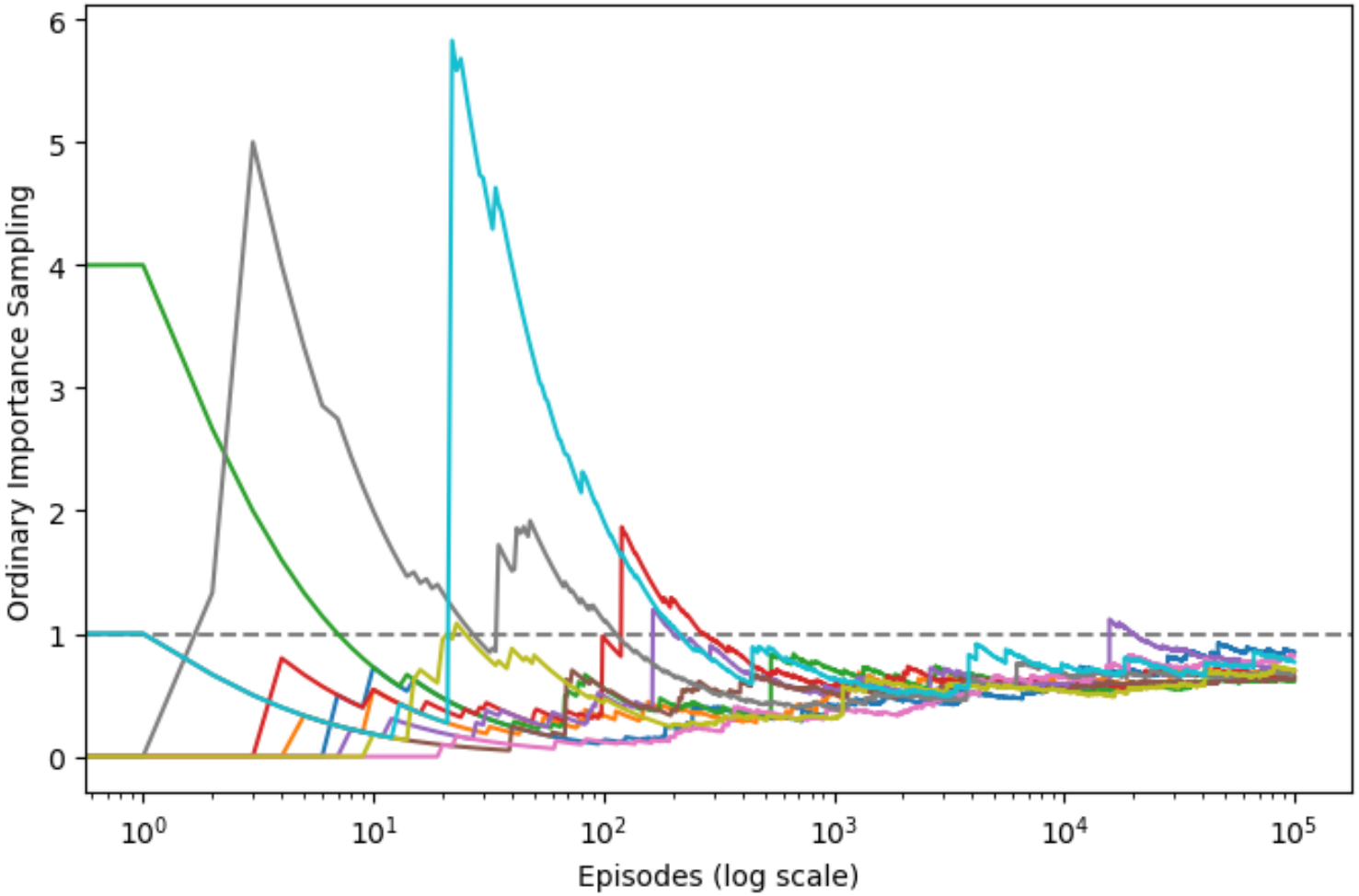
$$E[\rho_{t:T(t)-1}\, G_t \mid S_t = s] = v_\pi(s)$$

- The importance sampling ratio is:

$$\rho_{t:T(t)-1} = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}$$

- Since $\pi(a_R) = 0$, any episode that takes a_R has $\rho = 0$.
- If an episode follows the target policy, repeatedly taking a_L, the probability of that trajectory under:
  - Target policy: 1
  - Behavior policy: 0.5^n (where n is the number of times a_L is chosen)
- But the second moment $E[(\rho_{t:T(t)-1}\, G_t)^2]$ is not finite!

Example: Infinite Variance

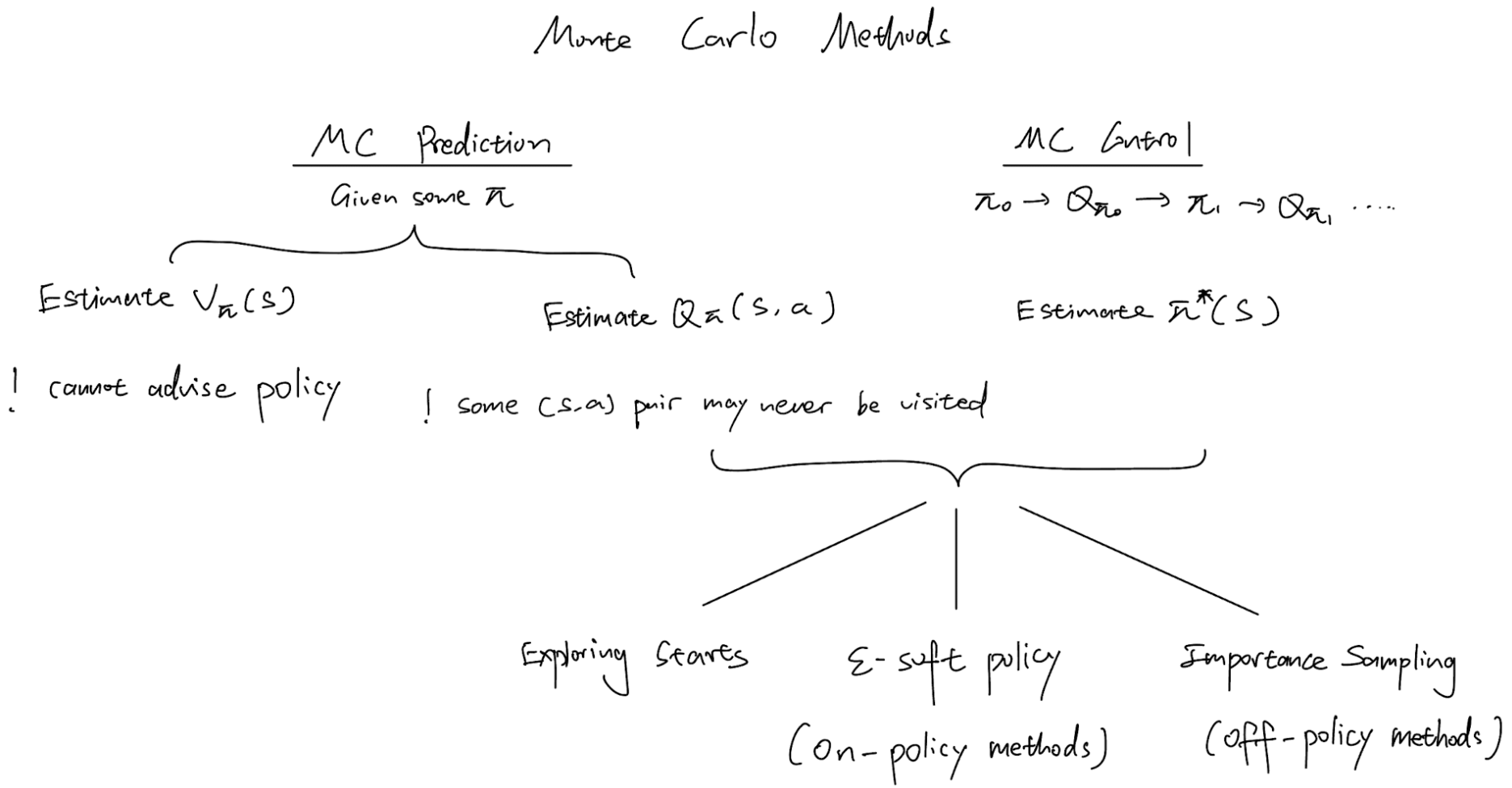


Ordinary vs. Weighted Importance Sampling

- Ordinary importance sampling is unbiased but has high variance.
- Weighted importance sampling is more stable and preferred in practice.

Feature	Ordinary IS	Weighted IS
Unbiased?	Yes	No (slightly biased)
Variance	High	Lower
Stability	Unstable	More stable
Convergence	Slower	Faster
Practical Use	Less reliable	Preferred in practice

Summary


Monte Carlo Methods
MC Prediction
Given some π
MC Control
π0 → Qπ0 → π1 → Qπ1 ....
Estimate Vπ(s)
Estimate Qπ(s, a)
Estimate π*(s)
! cannot advise policy
! some (s,a) pair may never be visited
Exploring Starts
ε-soft policy
(on-policy methods)
Importance Sampling
(off-policy methods)

Discussion

- In Monte Carlo methods, the estimates for each state are independent.
- The estimate for one state does not build upon the estimate of any other state.
- This is different from Dynamic Programming (DP), where state values depend on other states.
- Monte Carlo methods do not use bootstrapping, while *Temporal-Difference (TD) learning* in Chapter 6 will incorporate bootstrapping.

Thank you!